

What is claimed is:

1. A method comprising:

relocating a compiled code block associated with a software application;

wherein said relocating is performed responsive to hardware event information gathered during current execution of the software application; and

wherein said relocating is performed during current execution of the software application.

2. The method of claim 1, further comprising:

selecting the compiled code block responsive to occurrence of a trigger condition during current execution of the software application.

3. The method of claim 2, wherein:

the hardware event information indicates that the trigger condition has occurred during current execution of the software program.

4. The method of claim 2, wherein:

the trigger condition is a threshold number of hardware performance events.

5. The method of claim 4, wherein:

the trigger condition is a threshold number of instruction miss events.

1

1

2

1

1

2

1

1

2

3

1

1

2

3

1

1

2

3

1

1

- 6. The method of claim 2, wherein:

the trigger condition is a set of hardware criteria.

- 7. The method of claim 2, wherein:

the trigger condition is a set of hardware and software criteria.

- 8. The method of claim 1, further comprising:

selecting the compiled code block based on the compiled code block's resource
utilization during current execution of the software application.

- 9. The method of claim 8, wherein:

the hardware event information reflects the compiled code block's resource utilization
during current execution of the software program.

- 10. The method of claim 9, wherein:

the hardware event information further reflects the number of executed method calls
performed by the compiled code block.

- 11. The method of claim 9, wherein:

the hardware event information further reflects the number of times the compiled code block has been called during execution of the software program.

12. The method of claim 9, wherein:

the hardware event information further reflects the number of execution cycles consumed during execution of the compiled code block.

13. The method of claim 1, further comprising:

relocating a virtual method table associated with the software application during current execution of the software application.

14. The method of claim 13, wherein:

said hardware event information includes data miss information.

15. The method of claim 1, wherein:

said relocating is performed on an as-needed basis independent of garbage collection.

16. The method of claim 1, wherein said relocating further comprises:

moving the compiled code block to a new location within a code region; and

3 patching address references in the code region to reflect the new location of the compiled
4 code block.

1
1 17. The method of claim 16, wherein:

2 patching address references further comprises patching address references such that
3 invocation of the relocated compiled code does not generate a trap.

1
1 18. The method of claim 16, further comprising:

2 patching a call stack to reflect the new location.

1
1 19. The method of claim 16, further comprising:

2 patching a virtual method table to reflect the new location.

1
1 20. A method, comprising:

2 dynamically relocating a program element; and

3 invoking a just-in-time compiler to patch address references associated with the relocated
4 program element.

1
1 21. The method of claim 20, wherein:

2 the program element is a compiled code block.

1 22. The method of claim 20, wherein:

2 the program element is a virtual method table.

1

1 23. The method of claim 20, further comprising:

2 generating hardware event information during current execution of a software program

3 with which the program element is associated.

1

1 24. The method of claim 23, wherein dynamically relocating a program element

2 further comprises:

3 determining whether to relocate the program element based on the hardware event

4 information.

1

1 25. The method of claim 24, wherein determining whether to relocate the program

2 element further comprises:

3 determining whether the hardware event information indicates that a trigger condition

4 has been met.

1

1 26. The method of claim 24, wherein:

2 the program element is a compiled code block; and

3 determining whether to relocate the program element further comprises determining
4 whether calls to the compiled code block have generated at least a predetermined number
5 of instruction miss events.

1
1 27. The method of claim 24, wherein:

2 the program element is a virtual method table; and
3 determining whether to relocate the program element further comprises determining
4 whether accesses to the virtual method table have generated at least a predetermined
5 number of data miss events.

1
1 28. The method of claim 23, wherein dynamically relocating a program element
2 further comprises:

3 determining a new location for the relocated program element based on the hardware
4 event information.

1
1 29. The method of claim 23, wherein:

2 the hardware event information includes branch history information.

1
1 30. The method of claim 23, further comprising:

2 generating profile information based on the hardware event information.

1 31. The method of claim 30, wherein:

2 the profile information includes a call graph.

1 32. An article comprising:

2 a machine-readable storage medium having a plurality of machine accessible instructions,
3 which if executed by a machine, cause the machine to perform operations comprising:

4 dynamically relocating a program element; and

5 invoking a just-in-time compiler to patch address references associated with the
6 relocated program element.

1 33. The article of claim 32, wherein the storage medium has instructions, which if
2 executed by a machine, cause the machine to further perform operations comprising:

3 generating hardware event information during current execution of a software
4 program with which the program element is associated.

1 34. The article of claim 33, wherein the instructions that cause the machine to
2 dynamically relocate a program element further include instructions, which if executed by a
3 machine, cause the machine to further perform operations comprising:

4 determining whether to relocate the program element based on the hardware event
5 information.

1 35. The article of claim 33, wherein the instructions that cause the machine to
2 dynamically relocate a program element further include instructions, which if executed by a
3 machine, cause the machine to further perform operations comprising:

4 determining a new location for the relocated program element based on the hardware
5 event information.

1
1 36. The article of claim 34, wherein the instructions that cause the machine to
2 determine whether to relocate the program element further include instructions, which if
3 executed by a machine, cause the machine to perform operations comprising:

4 determining, based on the hardware event information, whether a trigger condition
5 has been met during current execution of the software program.

1
1 37. The article of claim 36, wherein the instructions that cause the machine to
2 determine whether a trigger condition has been met further include instructions, which if
3 executed by a machine, cause the machine to perform operations comprising:

4 determining whether calls to the program element have generated at least a
5 predetermined number of instruction miss events.

1
1 38. The article of claim 36, wherein the instructions that cause the machine to
2 determine whether a trigger condition has been met further include instructions, which if
3 executed by a machine, cause the machine to perform operations comprising:

4 determining whether accesses to the program element have generated at least a
5 predetermined number of data miss events.

1
1 39. The article of claim 36, wherein the instructions that cause the machine to
2 determine whether a trigger condition has been met further include instructions, which if
3 executed by a machine, cause the machine to perform operations comprising:

4 determining whether execution of the program element has utilized at least a
5 predetermined quantity of execution resources.

1
1 40. An apparatus, comprising:
2 a compiled code region to store compiled native codes; and
3 a code manager to dynamically modify layout of the compiled code region based on
4 hardware event feedback.

1
1 41. The apparatus of claim 40, wherein:
2 the hardware event feedback is generated during current execution of a software program.

1
1 42. The apparatus of claim 40, wherein:
2 the code manager is further to determine, based on the feedback, whether the layout of
3 the compiled code region should be modified.

1 43. The apparatus of claim 40, wherein:

2 the code manager is further to determine, based on the feedback, a new location for a
3 compiled code block.

1 44. The apparatus of claim 40, further comprising:

2 code manipulation logic to patch address references in the compiled code region.

1 45. The apparatus of claim 40, further comprising:

2 a virtual method table region.

1 46. The apparatus of claim 45, wherein:

2 the code manager is further to dynamically modify layout of the virtual method table
3 region based on hardware event feedback.

1 47. The apparatus of claim 46, wherein:

2 the code manager is further to determine, based on the feedback, whether the layout
3 of the virtual method table region should be modified.

1 48. The apparatus of claim 46, wherein:

the code manager is further to determine, based on the feedback, a new location for a virtual method table.

49. A system comprising:

a processor; and

a memory, wherein the memory further comprises:

a compiled code region;

a code manager to dynamically manage layout of the compiled code region; and

code manipulation logic to patch address references in the compiled code region.

50. The system of claim 49, further comprising:

a virtual method table region.

51. The system of claim 50, wherein:

the code manager is further to dynamically manage layout of the virtual method table region.

52. The system of claim 49, wherein:

the memory is a DRAM.

1 53. The system of claim 49, wherein:
2 the code manager is further to dynamically manage layout of the compiled code region
3 based on dynamic hardware event information.

1 54. The system of claim 53, wherein:
2 the code manager is further to dynamically manage layout of the compiled code region
3 based on dynamic hardware instruction miss information.

1 55. The system of claim 51, wherein:
2 the code manager is further to dynamically manage layout of the virtual method table
3 region based on dynamic hardware event information.

1 56. The system of claim 55, wherein:
2 the code manager is further to dynamically manage layout of the virtual method table
3 region based on dynamic hardware data miss information.